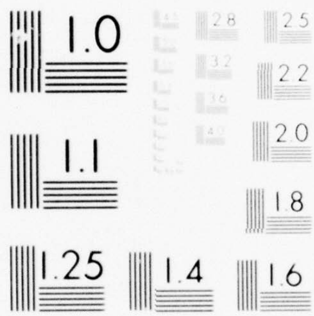
AD-A046 602 COLORADO STATE UNIV FORT COLLINS DEPT OF MATHEMATICS F/G 12/1
RESTRICTED RANGE ADAPTIVE CURVE FITTING.(U)
JUL 77 J A HULL, G D TAYLOR AFOSR-76-2878
UNCLASSIFIED AFOSR-TR-77-1258 NL

1 OF 1
AD
A046602

END
DATE
FILMED
12-77
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

AD A046602

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 18 AFOSR - TR- 77-1258 19	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	2
4. TITLE (and Subtitle): 6 RESTRICTED RANGE ADAPTIVE CURVE FITTING	5. TYPE OF REPORT & PERIOD COVERED: 9 Interim rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): 10 J. A. Hull [redacted] G. D. Taylor	8. CONTRACT OR GRANT NUMBER(s): 15 AFOSR-76-2878
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Colorado State University, Department of Mathematics, Fort Collins, CO 80523	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 16 2304 A3 17
11. CONTROLLING OFFICE NAME AND ADDRESS: AFOSR/NM, Bldg. 410, Bollins AFB, D.C. 20332	12. REPORT DATE: 11 July 1977
	13. NUMBER OF PAGES: 32
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 35p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Curve fitting, data fitting, adaptive curve fitting with user imposed constraints.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper ~~we~~ presents an algorithm for adaptively computing smooth piecewise polynomial approximations using restricted range uniform approximations. Also ~~We also present~~ several numerical examples are presented and ~~offer~~ offered suggestions for the effective use of this algorithm. ~~We have found~~ the algorithm is found to be effective for approximating a wide class of functions, either with or without significant levels of noise. Furthermore, since the user of this algorithm actually defines tolerance bands within which the approximation will lie, the algorithm allows

407344

16

20. the user a great deal of flexibility and control over the shape of the resulting approximations. A Fortran code of this algorithm is included in an appendix at the end of the paper.

AFOSR-TR- 77-1258

RESTRICTED RANGE ADAPTIVE CURVE FITTING

J. A. Hull and G. D. Taylor

Applied Technology, 645 Almanor, Sunnyvale, California 94086
Department of Mathematics, Colorado State University, Ft. Collins, Colorado 80523

ABSTRACT

In this paper we present an algorithm for adaptively computing smooth piecewise polynomial approximations using restricted range uniform approximations. We also present several numerical examples and offer suggestions for the effective use of this algorithm. We have found the algorithm to be effective for approximating a wide class of functions, either with or without significant levels of noise. Furthermore, since the user of this algorithm actually defines tolerance bands within which the approximation will lie, the algorithm allows the user a great deal of flexibility and control over the shape of the resulting approximations.

Send proofs to G. D. Taylor

[1] Research sponsored by the Air Force Office of Scientific Research, Air Force Systems, USAF, under grant no. 76-2878.

I. Introduction

Let X be a finite set of real points and let f be a function defined on X or let data be given in tabular form. In the case of data given in tabular form, say $\{(t_i, y_i)\}_{i=1}^{M}$, we shall set $X = \{t_i\}_{i=1}^{M}$ and define f on X by $f(t_i) = y_i$, $i = 1, \ldots, M$. In what follows we shall use this functional notation. Let $a = \min\{x: x \in X\}$ and $b = \max\{x: x \in X\}$. For any function g defined on X define $\|g\|_X = \max\{|g(x)|: x \in X\}$. Let SMTH and N be nonnegative integers supplied by the user, with N > SMTH. Let $u(x)$ and $\ell(x)$ be functions defined on X supplied by the user such that for each $x \in X$ we have $\ell(x) \leq f(x) \leq u(x)$ and $\ell(x) < u(x)$. In this setting our algorithm will calculate a piecewise polynomial approximation, p, to f, and a set of points $\{x_i\}_{i=0}^{k} \subset X$ with $a = x_0 < x_1 < \ldots < x_k = b$ such that p restricted to $[x_i, x_{i+1}]$ is a polynomial $p_i \in \Pi_{N-1} = \{q: q \text{ is a real algebraic polynomial of degree} \leq N-1\}$, p has SMTH continuous derivatives (on [a, b]) and for each $x \in X$, $\ell(x) \leq p(x) \leq u(x)$. By appropriately choosing $\ell(x)$ and $u(x)$ then, the user obtains an approximation meeting a given preselected (set of) tolerance(s).

II. The Algorithm

The algorithm begins by choosing $\tilde{x}_1$ to be the largest point in X such that

1) $[a, \tilde{x}_1] \cap X$ contains at least N+1 points and
2) There is a best restricted range uniform approximation p_1 from Π_{N-1} to f (with respect to the constraining curves $u(x)$ and $\ell(x)$) on $[a, \tilde{x}_1] \cap X$; that is, there exists $p_1 \in \Pi_{N-1}$ for which $\ell(x) \leq p_1(x) \leq u(x)$ holds for all $x \in [a, \tilde{x}_1] \cap X$ and $\|f - p_1\|_{[a,\tilde{x}_1]\cap X} = \inf\{ \|f - q\|_{[a,\tilde{x}_1]\cap X}: q \in \Pi_{N-1}$ and $\ell(x) \leq q(x) \leq u(x)$ for all $x \in [a,\tilde{x}_1]\cap X\}$.

If $\tilde{x}_1 = b$, then since p_1 is a piecewise polynomial meeting our requirements, we successfully terminate the algorithm. If no such $\tilde{x}_1$ exists, the algorithm fails and an appropriate error message is generated. Otherwise, if SMTH = 0 (i.e. we only require the approximation to be continuous) we choose the right endpoint of the first subinterval, x_1, to be $\tilde{x}_1$. If SMTH > 0, in order to add to the stability of the algorithm we (in general) choose x_1 by "backing off" from $\tilde{x}_1$ in the following manner.

We first examine the error curve $f(x) - p_1(x)$ and find those points $\xi_1, \xi_2, \ldots, \xi_\ell$ in $(a, \tilde{x}_1] \cap X$ at which relative extrema occur. We will choose one of the ξ_ν's to be x_1. The motivation for choosing x_1 in this manner is that in the continuous setting, if f is differentiable and ξ is an interior relative extreme point of $f(x) - p_1(x)$, then $f'(\xi) - p_1'(\xi) = 0$ so that the derivative of p_1 at ξ would match that of f at ξ. This guarantees that when we smoothly join the next piece of the approximation to p_1 at ξ, this next piece will closely follow the direction of f at least near ξ. If we merely joined the second piece to the first at $\tilde{x}_1$, no such guarantee can be made, and, in fact, severe oscillatory problems tend to set in. Our numerical experience indicates that the procedure of backing off from $\tilde{x}_i$ to a smaller x_i contributes very significantly toward the stability of the algorithm. To continue, let $\tilde{f}'(\xi_\nu)$ be the derivative of the centered quadratic interpolation of f evaluated at ξ_ν. We then choose x_1 to be the largest ξ_ν such that $|\tilde{f}'(\xi_\nu) - p_1'(\xi_\nu)| <$ EPS, where EPS is a tolerance which can be set by the user, or, if there does not exist such a ξ_ν, then we let x_1 be the largest ξ_ν at which $|f'(\xi_\nu) - p_1'(\xi_\nu)|$ is a minimum. (In our implementation of the algorithm, we do not in general consider all of the relative extreme points of $f(x) - p_1(x)$ in $[a, \tilde{x}_1] \cap X$, but only the largest N - SMTH - 1 of them.)

We continue by finding successive intervals $[x_1, x_2]$, $[x_2, x_3]$, ..., $[x_{m-1}, b]$ and corresponding polynomial approximations $p_2, p_3, \ldots, p_m \in \Pi_{N-1}$ to f so that $p_{\nu-1}^{(j)}(x_{\nu-1}) = p_\nu^{(j)}(x_{\nu-1})$ for j = 0, 1, ..., SMTH and $\ell(x) \le p_\nu(x) \le u(x)$ for every $x \in [x_{\nu-1}, x_\nu] \cap X$ for $\nu = 2, \ldots, m$, $x_m = b$. This is accomplished as follows:

Suppose we have found the subintervals $[a, x_1]$, $[x_1, x_2]$, ..., $[x_{i-2}, x_{i-1}]$, and the corresponding approximations $p_1, p_2, \ldots, p_{i-1}$. Assume further that $[x_{i-1}, b] \subset X$ contains at least max(2, N-SMTH) points. We now determine an x_i and a p_i meeting the above requirements. We begin by choosing $\tilde{x}_i \in X$ to be the largest point in X which satisfies

1) $[x_{i-1}, \tilde{x}_i]$ contains at least max(2, N-SMTH) points and
2) There exists a best restricted range uniform approximation, p_i, to f on $[x_{i-1}, \tilde{x}_i] \cap X$, subject to the constraint that $p_i^{(j)}(x_{i-1}) = p_{i-1}^{(j)}(x_{i-1})$, j = 0, 1, ..., SMTH.

If $\tilde{x}_i = b$, we set $x_i = \tilde{x}_i = b$, and the algorithm is successfully terminated. If no such $\tilde{x}_i$ exists, the algorithm fails and is terminated. Otherwise, we choose x_i completely analogous to our choice of x_1 above.

Finally, we consider the special case where $[x_{i-1}, b] \cap X$ contains fewer than max(2, N-SMTH) points. Specifically, we choose $\hat{x}_{i-1}$ to be a point in X closest to $(b - x_{i-2})/2$ which satisfies

1) $[\hat{x}_{i-1}, b] \cap X$ contains at least max(2, N-SMTH) points and
2) There exists a best restricted range approximation, p_2, to f from Π_{N-1} on $[\hat{x}_{i-1}, b] \cap X$ subject to the constraint that $p_i^{(j)}(\hat{x}_{i-1}) = p_{i-1}^{(j)}(\hat{x}_{i-1})$, j = 0, 1, ..., SMTH.

Again, if we can find such an $\hat{x}_{i-1}$ then we change x_{i-1} to $\hat{x}_{i-1}$, set $x_i = b$, and successfully terminate the algorithm. If we cannot find such an $\hat{x}_{i-1}$, the algorithm is terminated and an appropriate error message is generated.

Remark. In our implementation of this algorithm, the $\tilde{x}_i$ are chosen as follows. At each step of this iterative procedure we will let $\tilde{a}$ be the current largest point in X such that requirements (1) and (2) are satisfied on $[x_{i-1}, \tilde{a}] \cap X$, and we will let $\tilde{b}$ be the current smallest point in X such that $\tilde{b} > \tilde{a}$ and requirement (2) fails to be satisfied. We initialize this process by computing (or attempting to compute) the best restricted approximation on $[x_{i-1}, b] \cap X$ subject to the smoothness interpolatory constraints. If this approximation satisfies requirement (2), then we set $\tilde{x}_i = b$ and we are done. If the approximation fails to satisfy (2), we set $\tilde{b} = b$. Next, let $t = \min\{x \in X: [x_{i-1}, x] \cap X$ contains at least max(2, N-SMTH) points$\}$. If the approximation on $[x_{i-1}, t] \cap X$ fails to satisfy (2) then the algorithm cannot meet the desired accuracy and fails. Otherwise, we set $\tilde{a} = t$.

In general, we proceed as follows. We let $t = \inf\{x \in X: (\tilde{b}-\tilde{a})/2 \leq x < \tilde{b}\}$. If this set is empty, we set $t = \sup\{x \in X: \tilde{a} \leq x \leq (\tilde{b} - \tilde{a})/2\}$. If $t = \tilde{a}$ then this procedure has converged and we set $\tilde{x}_i = t = \tilde{a}$. Otherwise, we compute (or attempt to compute) the best restricted range approximation with interpolatory constraints on $[x_{i-1}, t] \cap X$. If this approximation satisfies (2) then we set $\tilde{a} = t$. If this approximation fails to satisfy (2) then we set $\tilde{b} = t$. We continue this process until $\tilde{b} - \tilde{a}$ is less than some user definable prescribed tolerance, at which point we accept $\tilde{a}$ as a good approximation to $\tilde{x}_i$ and terminate this procedure. We compute the $\hat{x}_i$ in a manner analogous to the above.

We have implemented a Remes-like algorithm to compute best restricted range uniform approximations with interpolatory constraints. See [2] for a complete discussion of this problem.

III. Numerical Results

By setting $u(x) = f(x) + TOL$ and $\ell(x) = f(x) - TOL$ for $x \in X$, where TOL is some positive, preselected tolerance, this algorithm simplifies to the best uniform approximation operator version of the algorithm given in [3]. For this reason we do not consider here any examples with restraining curves differing from the function being approximated by a constant. Indeed, the real value of this algorithm is that the tolerance we require our approximation to satisfy may vary from point to point. Consequently, where f is "nice" we can force our approximation to be close to f, and where f is "bad" we can relax our requirements, thereby obtaining an approximation which more closely reflects the character of f than can be obtained by selecting a fixed tolerance throughout the domain of approximation. In the case of experimental data which contain considerable levels of noise, the user can force the approximation to lie on the "believable" side of the data; often, more useful approximations can be obtained with this algorithm than can be obtained with (for example) the discrete L^2 version of the algorithm given in [3].

This algorithm has been implemented as a FORTRAN program running on Colorado State University's CDC CYBER 172 and CDC 6400. In the appendix we give a listing of the algorithm. As examples, we now present approximations to experimental data involving the release of bitumen and gas and oil from oil shale heated to a constant temperature as a function of time. Because relatively few data points were available (14-20), we filled in the gaps between the data points by discretizing (200-500 points) the linear interpolation of the original data using an algorithm which added (somewhat) more points in regions where the function

being approximated was complicated (i.e., radically changing slopes between data points) and fewer points in regions where the function is smooth. The advantage of this unequal spacing over equally spaced points is that in regions where the function being approximated is complicated, the procedure of "backing off" from $\tilde{x}_i$ to x_i becomes more effective by having more densely packed points. Also, the subintervals $[x_{i-1},\tilde{x}_i]$ may be chosen to be smaller (if needed in order to obtain a close enough approximation) while $[x_{i-1}, \tilde{x}_i] \cap X$ still contains at least max(2, N-SMTH) points. Only the original data points (indicated by "x") are shown in the following plots. In each case we chose N = 6, and SMTH = 2. The curves $\ell(x)$ and $u(x)$ were chosen by hand or by means of a simple algorithm at the original data points, and then they, too, were "filled in" using the same linear interpolation scheme as above. The TOL parameter listed on the plots is the largest tolerance allowed at any point throughout the approximation. This error is not in general reached.

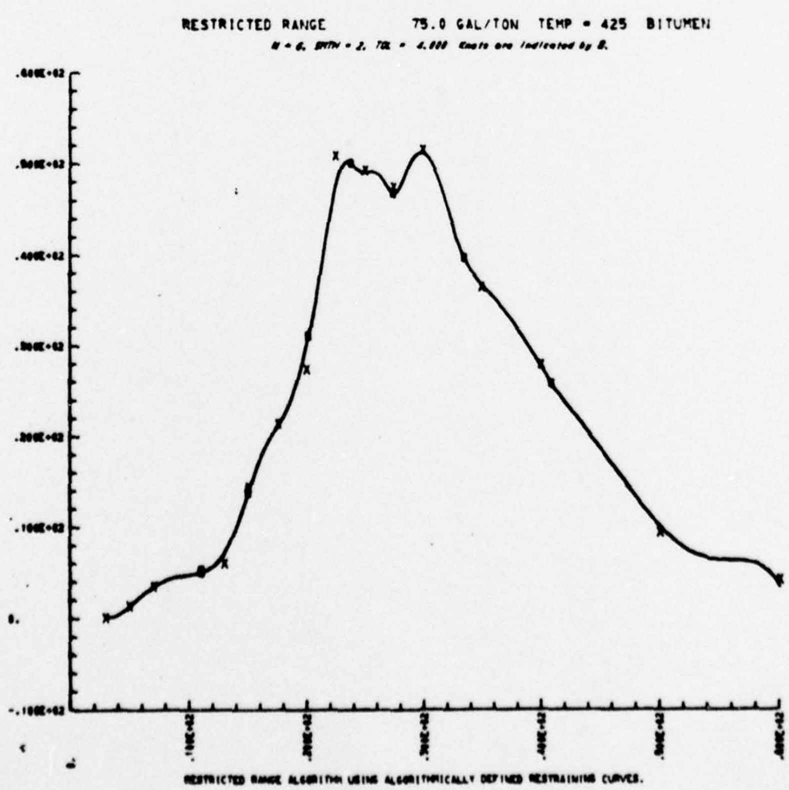


Figure 1

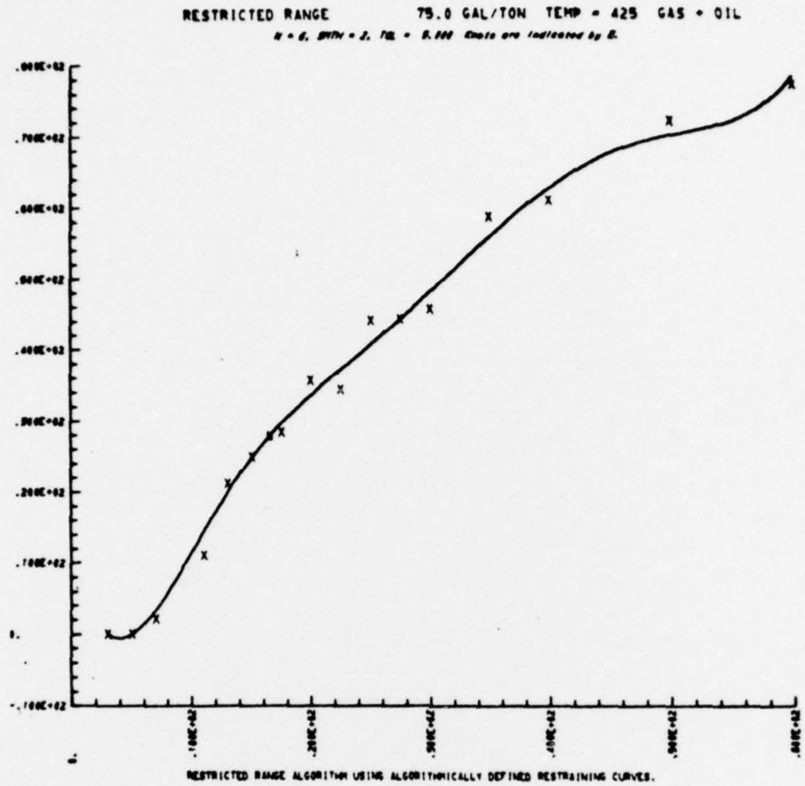


Figure 2

Although the maximum tolerance in both of these examples is fairly large, the tolerances throughout most of these intervals of approximation were on the order of .15 to 1.5. That is, we required fairly close agreement with the function being approximated except at the "bad" points. As an example of what can be done by appropriately choosing the restraining curves, we chose a band containing the above data and computed restraining curves based on this band as the following plot shows. The curve in the center is the resulting approximation.

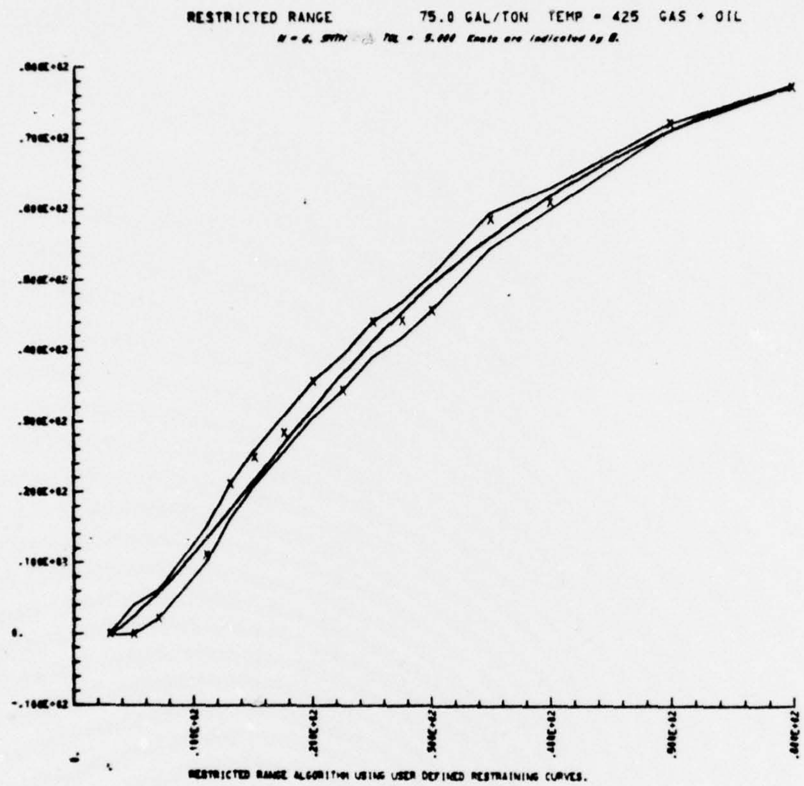


Figure 3

For clarity, we repeated the above plot but without the restraining curves.

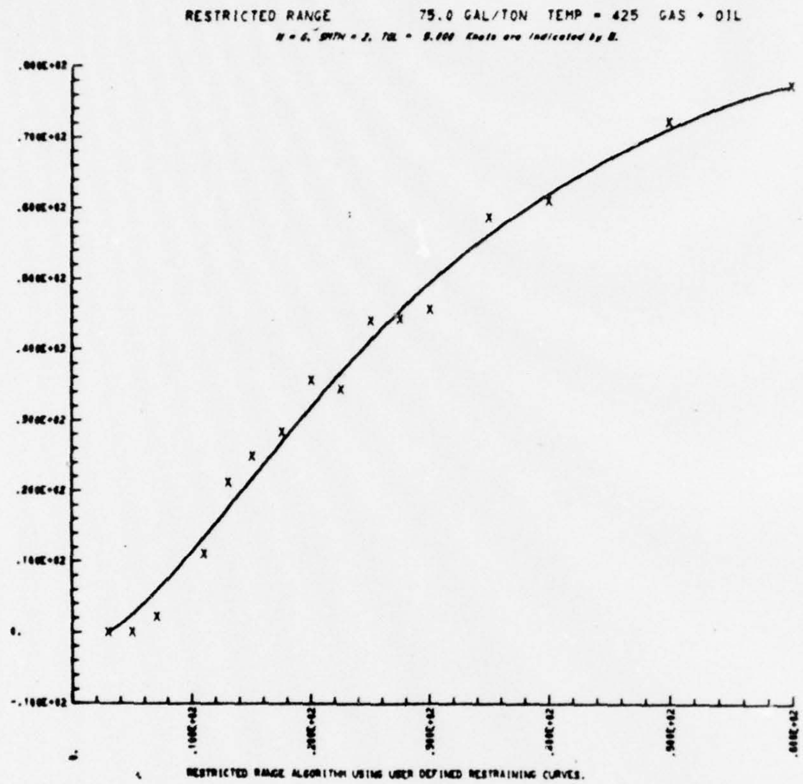


Figure 4

For additional examples using this algorithm and a similar algorithm using best L^2 approximations, see [1].

By appropriately setting the restraining curves the user can, to a large extent, determine the shape of the approximation. The most effective way of determining such restraining curves seems to be trial and error. Ideally, these restraining curves would be determined in an interactive setting using a graphics terminal with a pen light as follows. First one would make a rough initial guess at what the restraining curves should be (using some simple algorithm or otherwise), then allow the

algorithm to compute the first piece of the approximation. One would then display the data, the current approximation and the current restraining curves and modify the restraining curves on the relevant subinterval as desired so that when this first piece is recomputed using the updated restraining curves, it behaves as desired. After the user is satisfied with the first piece, he would repeat the above strategy on each successive subinterval as they are determined by the algorithm.

REFERENCES

[1] M. Andrews, J.A. Hull and G. D. Taylor, Adaptive curve fittings for chemical processes, to appear.

[2] B.L. Chalmers, The Remez Excnage Algorithm for Approximation with Linear Restrictions, Trans. Amer. Math. Soc., 222(1976), 103-131.

[3] J.A. Hull and G.D. Taylor, Adaptive Curve Fitting, to appear.

APPENDIX

Here we give a listing with driver, sample input and output which corresponds to Figure 1. The sample input is located after the listing of the code and prior to the sample output. This example uses the algorithmically defined restraining curves. In addition, instructions for user defined restraining curves are given in the sample input section.

```
      PROGRAM DRIVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      LOGICAL ERROR
      COMMON XTABLE(500),FTABLE(500),DUMMY(1582),FU(500),FL(500)
      INTEGER SMTH
      READ(5,50)IOPT
      READ (5,100) N,SMTH,TOL,SMLTOL
      IF(IOPT.EQ.0)WRITE(6,200)
      IF(IOPT.EQ.1)WRITE(6,300)
      WRITE(6,400)N,SMTH,TOL
      MAXNUM=0
   10 MAXNUM=MAXNUM+1
      READ (5,150) XTABLE(MAXNUM),FTABLE(MAXNUM),FU(MAXNUM),FL(MAXNUM)
      IF (EOF(5).EQ.0.0) GO TO 10
      MAXNUM=MAXNUM-1
      IF(IOPT.EQ.1)CALL SETRNG(MAXNUM,SMLTOL,TOL)
      CALL LINEAR (MAXNUM,300,MAXNUM,1.125)
      CALL RPACF (N,SMTH,MAXNUM,ERROR)
   50 FORMAT(I1)
  100 FORMAT(2I5,2F10.5)
  150 FORMAT (4F10.5)
  200 FORMAT(1H1,5X,60HRESTRICTED RANGE ALGORITHM--USER DEFINED RESTRAIN
     $ING CURVES.)
  300 FORMAT(1H1,5X,71HRESTRICTED RANGE ALGORITHM--ALGORITHMICALLY DEFIN
     $ED RESTRAINING CURVES.)
  400 FORMAT(1H0,5X,17HNUMBER OF COEFFS=,I2,1H,,I2,25H CONTINUOUS DERIVS
     $, TOL= ,F7.5)
      END


      SUBROUTINE LINEAR (OLDMAX,NAPROX,NTRUE,R)
C
C  THIS SUBROUTINE FILLS IN BETWEEN THE ORIGINAL DATA POINTS WITH
C  ABOUT NAPROX LINEARLY INTERPOLATED DATA POINTS.  THE SPACING USED
C  DEPENDS UPON THE DATA.  FOR *SMOOTH* DATA, I.E., DATA SUCH THAT CON-
C  SECUTIVE LINE SEGMENTS JOINING THE DATA POINTS DIFFER ONLY SLIGHTLY
C  IN SLOPE, EQUISPACED POINTS ARE USED.  FOR MORE COMPLEX DATA, THE
C  SPACING IS SUCH THAT MORE POINTS ARE CONCENTRATED IN AREAS WHERE
C  THE FUNCTION BEING APPROXIMATED IS COMPLICATED AND FEWER WHERE IT
C  IS SIMPLE.  THE MAXIMUM SPACING BETWEEN POINTS IS DELTA*R, WHERE
C  DELTA IS THE SPACING WHICH WOULD RESULT IF WE USED NAPROX EQUI-
C  SPACED POINTS, AND R IS A CONSTANT CHOSEN BY THE USER WHICH IS
C  GREATER THAN OR EQUAL TO 1.0.  (NOTE WHEN R IS SET TO 1.0, EQUI-
C  SPACING OF POINTS OCCURS REGARDLESS OF THE NATURE OF THE DATA.)
C  AS THE ORIGINAL DATA SET IS INCLUDED AS A SUBSET OF THE NEW DATA
C  SET,THE NUMBER OF POINTS IN THE NEW DATA SET IS BETWEEN NAPROX
C  AND NAPROX+OLDMAX.  THIS TRUE VALUE IS RETURNED TO THE USER IN
C  NTRUE.
C
      LOGICAL EQSPCD
      REAL MINVAL,M
      INTEGER OLDMAX,OLDM1
      COMMON XTABLE(500),FTABLE(500),XSTR(316),FSTR(316),FUSTR(316),FLST
     1R(317),QSUB(317),FU(500),FL(500)
      Q(I)=((FSTR(I+1)-FSTR(I))/(XSTR(I+1)-XSTR(I))-(FSTR(I)-FSTR(I-1))/
     1(XSTR(I)-XSTR(I-1)))
      AREA(I)=.5*(QSUB(I+1)+QSUB(I))*(XSTR(I+1)-XSTR(I))
      DATA EPS,OMEGA/1.0E-8,1.0E-5/
      DO 10 I=1,OLDMAX
         XSTR(I)=XTABLE(I)
         FSTR(I)=FTABLE(I)
         FUSTR(I)=FU(I)
         FLSTR(I)=FL(I)
```

```
10 CONTINUE
   EQSPCD=.TRUE.
   IF(R.EQ.1.0)GO TO 55
   OLDM1=OLDMAX-1
   SML=ABS(Q(2))
   DO 20 I=2,OLDM1
      TEMP=ABS(Q(I))
      QSUB(I)=TEMP
      IF (TEMP.LT.SML) SML=TEMP
20 CONTINUE
   QSUB(1)=0.0
   QSUB(OLDMAX)=0.0
   DO 30 I=2,OLDM1
30 QSUB(I)=QSUB(I)-SML
   MINIVL=1
   MINVAL=QSUB(2)
   TOTAR=MINVAL*.5*(XSTR(2)-XSTR(1))
   DO 40 I=2,OLDM1
      TEMP=QSUB(I)+QSUB(I+1)
      TOTAR=TOTAR+.5*TEMP*(XSTR(I+1)-XSTR(I))
      IF (TEMP.GE.MINVAL) GO TO 40
      MINVAL=TEMP
      MINIVL=I
40 CONTINUE
   DELTA=(XSTR(OLDMAX)-XSTR(1))/FLOAT(NAPROX-1)
   DLTMAX=DELTA*R
   IF(TOTAR.LT.OMEGA)GO TO 55
   EQSPCD=.FALSE.
   M=ABS(QSUB(MINIVL+1)-QSUB(MINIVL))/(XSTR(MINIVL+1)-XSTR(MINIVL))
   H=(TOTAR/FLOAT(NAPROX-1)-.5*M*DLTMAX*DLTMAX)/(DLTMAX-DELTA)
   IF (H.LT.0.0) H=0.0
   DO 50 I=1,OLDMAX
      QSUB(I)=QSUB(I)+H
50 CONTINUE
   C=(TOTAR+H*(XSTR(OLDMAX)-XSTR(1)))/FLOAT(NAPROX-1)
55 K=0
   I=0
   XX=XSTR(1)
60 I=I+1
   IF (XX.LT.XSTR(K+1)) GO TO 70
   K=K+1
   XX=XSTR(K)
   XTABLE(I)=XX
   FTABLE(I)=FSTR(K)
   FU(I)=FUSTR(K)
   FL(I)=FLSTR(K)
   IF (K.EQ.OLDMAX) GO TO 110
   DX=XSTR(K+1)-XSTR(K)
   SLF=(FSTR(K+1)-FSTR(K))/DX
   SLFU=(FUSTR(K+1)-FUSTR(K))/DX
   SLFL=(FLSTR(K+1)-FLSTR(K))/DX
   M=(QSUB(K+1)-QSUB(K))/DX
   IF(.NOT.EQSPCD)GO TO 80
   XX=XX+DELTA
   GO TO 60
70 XTABLE(I)=XX
   DIFF=XX-XSTR(K)
   FTABLE(I)=FSTR(K)+DIFF*SLF
   FU(I)=FUSTR(K)+DIFF*SLFU
   FL(I)=FLSTR(K)+DIFF*SLFL
   IF(.NOT.EQSPCD)GO TO 80
   XX=XX+DELTA
```

```
      GO TO 60
   80 QOFXX=M*(XX-XSTR(K))+QSUB(K)
      IF (ABS(M).LT.EPS) GO TO 90
      TEMP=QOFXX**2+2.0*M*C
      IF (TEMP.LT.0.0) GO TO 100
      XX=XX+(-QOFXX+SQRT(TEMP))/M
      GO TO 60
   90 XX=XX+C/QOFXX
      GO TO 60
  100 XX=XSTR(K+1)
      GO TO 60
  110 NTRUE=I
      RETURN
C
      END


      SUBROUTINE SETRNG (MAXNUM,MINTOL,MAXTOL)
C
C  THIS SUBROUTINE USES CONVEX COMBINATIONS OF MINTOL AND MAXTOL TO SET
C  THE FU AND FL TOLERANCES AT EACH DATA POINT DEPENDING ON THE COMP-
C  LEXITY OF THE FUNCTION BEING APPROXIMATED.  THAT IS, WHERE THE FUN-
C  CTION IS SMOOTHEST, THE TOLERANCES WILL BE CLOSE TO (BUT AT LEAST AS
C  BIG ON AT LEAST ONE SIDE) AS MINTOL, AND WHERE THE APPROXIMATION IS
C  COMPLICATED THE APPROXIMATION WILL BE CLOSE TO (BUT NO BIGGER THAN)
C  MAXTOL ON AT LEAST ONE SIDE.  FOR DATA WHICH IS *SMOOTH*, AS DE-
C  SCRIBED IN SUBROUTINE LINEAR, THE TOLERANCES BECOME SOMEWHAT LESS
C  SIGNIFICANT, AND ARE SET TO MAXTOL AT ALL DATA POINTS.  THIS
C  ROUTINE FREES THE USER FROM HAVING TO CHOOSE AN INITIAL BAND OF
C  TOLERANCES--IT IS NOT NECESSARILY INTENDED TO BE A TOTALLY AUTO-
C  MATIC TOLERANCE BAND SELECTOR FOR ANY ARBITRARY FUNCTION.  EXPER-
C  IMENTING WITH USER SUPPLIED TOLERANCES MOST OFTEN RESULTS IN MORE
C  DESIRABLE FITS.
C
      REAL MINTOL,MAXTOL
      COMMON XTABLE(500),FTABLE(1582),QSTR(500),FU(500),FL(500)
      DATA OMEGA/1.0E-5/
      Q(I)=((FTABLE(I+1)-FTABLE(I))/(XTABLE(I+1)-XTABLE(I))-(FTABLE(I)-F
     1TABLE(I-1))/(XTABLE(I)-XTABLE(I-1)))*(XTABLE(I+1)-XTABLE(I-1))
      MAXM1=MAXNUM-1
      QAVE=Q(2)
      QSTR(2)=QAVE
      BIG=ABS(QAVE)
      SML=BIG
      DO 10 I=3,MAXM1
         TEMP=Q(I)
         QSTR(I)=TEMP
         QAVE=QAVE+TEMP
         TEMP=ABS(TEMP)
         IF (TEMP.GT.BIG) BIG=TEMP
         IF (TEMP.LT.SML) SML=TEMP
   10 CONTINUE
      DIFF=BIG-SML
      IF(DIFF.LT.OMEGA)GO TO 40
      QAVE=QAVE/FLOAT(MAXNUM-2)
      DO 30 I=2,MAXM1
         TEMP=ABS(QSTR(I))
         TOLI=MINTOL*(BIG-TEMP)/DIFF+MAXTOL*(TEMP-SML)/DIFF
         TSUB=TOLI*(BIG-TEMP)/DIFF
         IF (QSTR(I).LT.0.0) GO TO 20
         FU(I)=TOLI
         FL(I)=TSUB
```

```
      GO TO 30
20    FU(I)=TSUB
      FL(I)=TOLI
30 CONTINUE
   FU(1)=FU(2)
   FL(1)=FL(2)
   FU(MAXNUM)=FU(MAXM1)
   FL(MAXNUM)=FL(MAXM1)


   RETURN
40 DO 50 I=1,MAXNUM
      FU(I)=MAXTOL
      FL(I)=MAXTOL
50 CONTINUE
   RETURN
C
   END

   SUBROUTINE RRACF (N,SMTH,MAXNUM,ERROR)
C
C  THIS SUBROUTINE ADAPTIVELY COMPUTES A PIECEWISE POLYNOMIAL APPROXI-
C  MATION OF DEGREE N-1 TO THE FUNCTION STORED IN THE ARRAYS XTABLE AND
C  FTABLE WITH SMTH CONTINUOUS DERIVATIVES HAVING THE PROPERTY THAT FOR
C  1 .LE. I .LE. MAXNUM (SEE BELOW) THE VALUE OF THE APPROXIMATION
C  EVALUATED AT XTABLE(I) LIES BETWEEN (FTABLE(I) - FL(I)) AND
C  (FTABLE(I) + FU(I)), WHERE THE ARRAYS FL AND FU CONTAIN THE DESIRED
C  TOLERANCE REQUIRED OF THE APPROXIMATION BELOW THE CURVE AND ABOVE THE
C  CURVE (RESPECTIVELY) AT EACH OF THE (MAXNUM) POINTS BEING APPROXI-
C  MATED.
C
C  THE PARAMETERS ARE AS FOLLOWS--
C
C       N - THE NUMBER OF COEFFICIENTS OF EACH POLYNOMIAL PIECE.  I.E.
C          ONE MORE THAN THE DEGREE OF THE PIECEWISE POLYNOMIAL APPROX.
C          AS THE ARRAYS ARE CURRENTLY DIMENSIONED, IT IS ASSUMED THAT
C          N IS NO BIGGER THAN 17.
C
C       SMTH - THE NUMBER OF CONTINUOUS DERIVATIVES DESIRED OF THE
C          APPROXIMATION.  SMTH MUST NOT BE GREATER THAN N-2.  IF ONLY
C          CONTINUITY OF THE APPROXIMATION IS REQUIRED, SET SMTH = 0.
C
C       MAXNUM - THE NUMBER OF POINTS ACTUALLY STORED IN THE ARRAYS
C          XTABLE, FTABLE, FU, AND FL.  (AS THESE ARRAYS ARE CURRENTLY
C          DIMENSIONED, MAXNUM MUST BE LESS THAN 500.  IF ONE WISHES TO
C          COMPUTE APPROXIMATIONS TO FUNCTIONS WITH MORE THAN 500
C          POINTS, ONE CAN EASILY MODIFY THIS PROGRAM TO CONTINUOUSLY
C          READ IN MORE POINTS AFTER ROOM IS MADE IN THESE STORAGE
C          ARRAYS BY HAVING COMPLETED SEVERAL SUBINTERVALS.)
C
C       ERROR - A LOGICAL VALUE SET TO .TRUE. IF AN ERROR OCCURS IN THE
C          PROGRAM (AN APPROPRIATE ERROR MESSAGE WILL ALSO BE PRINTED)
C          AND SET TO .FALSE. OTHERWISE.
C
C       BLANK COMMON PROVIDES THE REMAINDER OF THE INPUT.  IT IS ASSUMED
C       THAT THE FIRST 500 WORDS OF BLANK COMMON CONTAIN THE TABLE OF X
C       VALUES (XTABLE), THE NEXT 500 WORDS THE TABLE OF FUNCTION VALUES
C       (FTABLE), THE NEXT TWO WORDS ARE USED INTERNALLY THROUGHOUT THE
C       THE PROGRAM, AND THE NEXT 1080 WORDS IS AN ARRAY (CSTORE(18,60))
C       CONTAINING THE COMPUTED COEFFICIENTS AND THE KNOTS--I.E.
C       CSTORE(I,INT) IS THE COEFFICIENT OF THE I-1ST DEGREE TERM IN SUB-
```

```
C     INTERVAL NUMBER INT.  CSTORE(N+1,INT) IS THE LEFT END POINT OF    RRACF430
C     SUBINTERVAL INT.  THE NEXT ARRAY IS USED INTERNALLY TO STORE THE  RRACF440
C     ERRORS OF APPROXIMATION OCCURING AT EACH DATA POINT DURING THE    RRACF450
C     THE REMES ALGORITHM AND IT IS ALSO USED AS A SCRATCH STORAGE AR-  RRACF460
C     RAY THROUGHOUT THE PROGRAM.  THE LAST TWO ARRAYS IN BLANK COMMON  RRACF470
C     (FU AND FL) ARE THE ARRAYS IN WHICH THE USER SETS THE TOLERANCE   RRACF480
C     HE REQUIRES ABOVE AND BELOW THE CURVE AT EACH OF THE MAXNUM       RRACF490
C     POINTS AS DESCRIBED ABOVE.  FU AND FL ARE EACH DIMENSIONED 500    RRACF500
C     WORDS LONG.                                                       RRACF510
C                                                                       RRACF520
C  THE COEFFICIENTS AND SUBINTERVAL ENDPOINTS ARE PRINTED OUT AS THEY   RRACF530
C  ARE COMPUTED.  ALSO THE FUNCTION EVAL IS AVAILABLE TO THE USER TO    RRACF540
C  EVALUATE THE APPROXIMATION AT ANY POINT WITHIN THE ENTIRE INTERVAL.  RRACF550
C                                                                       RRACF560
      LOGICAL LAST,ERROR,DONE,ABORT                                     RRACF570
      INTEGER SMTH                                                      RRACF580
      DIMENSION C(18)                                                   RRACF590
      COMMON XTABLE(1000),LCTNLE,LCTNRE,DUM1(2580)                      RRACF600
      COMMON /SCALAR/ NPLUS0,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT   RRACF610
C                                                                       RRACF620
      NRSMTH=-1                                                         RRACF630
      ERROR=.FALSE.                                                     RRACF640
      DONE=.FALSE.                                                      RRACF650
      ABORT=.FALSE.                                                     RRACF660
      NPLUS0=N                                                          RRACF670
      NPLUS1=N+1                                                        RRACF680
C                                                                       RRACF690
C  IN THE FIRST SUBINTERVAL THERE ARE NO INTERPOLATORY CONSTRAINTS--    RRACF700
C  CONSEQUENTLY, WE SET NLSMTH=-1.                                      RRACF710
C                                                                       RRACF720
      NLSMTH=-1                                                         RRACF730
      NX=NPLUS1-2-NLSMTH-NRSMTH                                         RRACF740
      NXM1=NX-1                                                         RRACF750
      LNGTH=NX+1                                                        RRACF760
C                                                                       RRACF770
C  WE INITIALLY TRY AS MUCH OF THE CURRENT REMAINING PORTION OF THE     RRACF780
C  WHOLE INTERVAL AS POSSIBLE AS AN INITIAL GUESS FOR EACH SUCCESSIVE   RRACF790
C  SUBINTERVAL.  LCTNLE IS THE LOCATION (IN THE ARRAY XTABLE) OF THE    RRACF800
C  LEFT END POINT OF THE CURRENT SUBINTERVAL, LCTNRE IS THE LOCATION OF RRACF810
C  THE RIGHT END POINT.                                                 RRACF820
C                                                                       RRACF830
      LCTNLE=1                                                          RRACF840
      LCTNRE=MAXNUM                                                     RRACF850
      DO 20 INTNUM=1,60                                                 RRACF860
         NINT=INTNUM                                                    RRACF870
C                                                                       RRACF880
C  SUBROUTINE COMPUT FINDS THE LARGEST SUBINTERVAL OF (XTABLE(LCTNLE),  RRACF890
C  XTABLE(LCTNRE)) WITH LEFT END POINT XTABLE(LCTNLE) SUCH THAT THE BEST RRACF900
C  APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL THE CONSTRAINTS.  THE RRACF910
C  RIGHT END POINT IS ≠BACKED OFF≠ TO THE LAST INTERIOR EXTREME POINT   RRACF920
C  OF F-P TO ADD TO THE STABILITY OF THE ALGORITHM.  THE LOCATION OF    RRACF930
C  THIS RIGHT END POINT IS STORED IN LCTNRE.  IF LCTNRE=MAXNUM (I.E.    RRACF940
C  WE ARE DONE), CONTROL IS PASSED TO LINE 40.  IF NO SUCH SUBINTERVAL  RRACF950
C  CAN BE FOUND, CONTROL IS PASSED TO LINE 50.  IF THERE ARE FEWER THAN RRACF960
C  LNGTH POINTS FROM LCTNRE TO MAXNUM, LAST IS SET TO .TRUE. AND THE    RRACF970
C  SPECIAL CASE SUBROUTINE LSTINT IS CALLED.                            RRACF980
C                                                                       RRACF990
         CALL COMPUT (C,LNGTH,MAXNUM,LAST,DONE,ABORT)                   RRAC1000
         IF (ABORT) GO TO 50                                            RRAC1010
         IF (DONE) GO TO 40                                             RRAC1020
         IF (INTNUM.GT.1) GO TO 10                                      RRAC1030
         NLSMTH=SMTH                                                    RRAC1040
```

```
      NX=NPLUS1-2-NLSMTH-NRSMTH
      NXM1=NX-1
      LNGTH=NX+1
   10 IF (LAST) GO TO 30
C
C  SUBROUTINE STORE STORES THE COEFFICIENTS FOR THIS SUBINTERVAL IN THE
C  ARRAY CSTORE.  IT ALSO PRINTS OUT THE COEFFICIENTS AND THE ERROR OF
C  APPROXIMATION ON THIS SUBINTERVAL.  SUBROUTINE SETP(C,X,K) STORES THE
C  VALUE OF THE POLYNOMIAL DETERMINED BY THE COEFFICIENTS IN THE ARRAY
C  C AND ITS FIRST K DERIVATIVES AT THE POINT X IN THE ARRAY PPRIME.
C
         CALL STORE (C,LCTNLE,LCTNRE)
         CALL SETP (C,XTABLE(LCTNRE),NLSMTH)
         LCTNLE=LCTNRE
         LCTNRE=MIN0(MAXNUM,MAXINT+LCTNLE-1)
   20 CONTINUE
      WRITE (6,60) NINT
      ERROR=.TRUE.
      RETURN
   30 CALL LSTINT (C,LNGTH,MAXNUM,ABORT)
      IF (ABORT) GO TO 50
   40 CALL STORE (C,LCTNLE,LCTNRE)
      RETURN
   50 WRITE (6,70)
      ERROR=.TRUE.
      RETURN
C
   60 FORMAT (1H0,39(2H* ),1H*,/,2H0*,12X,37HTHIS APPROXIMATION REQUIRES
     1 MORE THAN,I3,13H SUBINTERVALS,12X,1H*,/,2H0*,28X,20H--PROGRAM ABO
     2RTING--,20X,1H*,/,1H0,39(2H* ),1H*)
   70 FORMAT (1H0,39(2H* ),1H*,/,2H0*,15X,46HTHE ALGORITHM CANNOT MEET T
     1HE DESIRED ACCURACY,16X,1H*,/,2H0*,28X,20H--PROGRAM ABORTING--,29X
     2,1H*,/,1H0,39(2H* ),1H*)
C
      END


      SUBROUTINE COMPUT (C,LNGTH,MAXNUM,LAST,DONE,ABORT)
C
C  THIS SUBROUTINE FINDS THE LARGEST SUBINTERVAL AND THE BEST APPROX-
C  IMATION TO F ON THIS SUBINTERVAL SUCH THAT THE APPROXIMATION MEETS
C  THE DESIRED ERROR TOLERANCE ON THE SUBINTERVAL.
C
      INTEGER A,B
      LOGICAL LAST,OK,DONE,ABORT,TOOBIG
      REAL C(18)
      COMMON XTABLE(1000),LCTNLE,LCTNRE,CSTORE(18,60),CDERIV(500)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /COMP/ LCTNX(18)
C
C  WE ASSUME THAT WE ARE CLOSE ENOUGH TO THE TRUE LARGEST SUBINTERVAL
C  RIGHT END POINT WHEN WE KNOW THAT OUR APPROXIMATION TO THE TRUE RIGHT
C  END POINT IS WITHIN ETA OF THE TRUE END POINT.
C
      DATA ETA/.08/
C
      LITTLE=LCTNLE+LNGTH-1
      A=0
      LAST=.FALSE.
   10 NUMPTS=LCTNRE-LCTNLE+1
C
C  IF THERE DOES NOT EXIST A BEST RESTRICTED RANGE APPROXIMATION ON THE
```

```
C  CURRENT SUBINTERVAL, CONTROL IS PASSED TO LINE 30.
C
      CALL REMES (C,LCTNX,TOOBIG,ABORT)
      IF (ABORT) RETURN
      IF (TOOBIG) GO TO 30
      IF (LCTNRE.LT.MAXNUM) GO TO 20
      DONE=.TRUE.
      RETURN
C
C  A IS THE CURRENT LARGEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL SATIFIES ALL CONSTRAINTS.
C
   20 A=LCTNRE
      IF ((XTABLE(B)-XTABLE(A).GT.ETA).AND.(B-A.GT.1)) GO TO 40
      GO TO 50
C
C  B IS THE CURRENT SMALLEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL FAILS TO SATISFY THE CON-
C  STRAINTS.
C
   30 B=LCTNRE
   40 NEWTRY=(A+B)/2+1
      IF (NEWTRY.EQ.B) NEWTRY=NEWTRY-1
      IF (NEWTRY.LT.LITTLE) NEWTRY=LITTLE
      IF (NEWTRY.EQ.LCTNRE) GO TO 50
      LCTNRE=NEWTRY
      GO TO 10
C
C  IF A IS STILL 0, THEN NO SUBINTERVAL WITH AT LEAST LNGTH POINTS
C  WILL WORK, SO THE ALGORITHM IS TERMINATED.
C
   50 IF (A.NE.0) GO TO 60
      ABORT=.TRUE.
      RETURN
C
C  SINCE NEWTRY IS ALWAYS STRICTLY LESS THAN THE CURRENT B, IF NEWTRY=
C  LCTNRE, AND A IS NOT STILL 0, NEWTRY=A, WHICH IS A POINT WHICH SAT-
C  ISFIES ALL REQUIREMENTS.  WE NOW BACK THE RIGHT ENDPOINT OFF TO THE
C  BEST INTERIOR EXTREME POINT OF F-P TO ADD TO THE STABILITY OF THE AL-
C  GORITHM.
C
   60 DO 70 I=1,N
         CDERIV(I)=C(I)
         CSTORE(I,NINT)=C(I)
   70 CONTINUE
      NDUMMY=N
      CALL DERIV (CDERIV,NDUMMY)
      I=NX
      LCTNRE=LCTNX(I)
      NEWTRY=LCTNRE
      SMLL=CMPR(CDERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)
      IF (OK) GO TO 100
   80 I=I-1
      IF (I.EQ.0) GO TO 100
      NEWTRY=LCTNX(I)
      IF (NEWTRY.LT.LNGTH) GO TO 100
      TEMP=CMPR(CDERIV,NDUMMY,NEWTRY,LCTNLE-1,OK)
      IF (.NOT.OK) GO TO 90
      LCTNRE=NEWTRY
      GO TO 100
   90 IF (TEMP.GE.SMLL) GO TO 80
      SMLL=TEMP
```

```
      LCTNRE=NEWTRY
      GO TO 80
  100 LCTNRE=LCTNLE+LCTNRE-1
      IF (MAXNUM-LCTNRE+1.LT.LNGTH) LAST=.TRUE.
      RETURN
C
      END


      SUBROUTINE LSTINT (C,LNGTH,MAXNUM,ABORT)
C
C  THIS SUBROUTINE HANDLES THE SPECIAL CASE OF FINDING A SUBINTERVAL
C  AND A BEST APPROXIMATION ON THAT SUBINTERVAL WHEN THERE ARE TOO
C  FEW REMAINING POINTS FOR COMPUT TO WORK.
C
      INTEGER OLDLE,OLDRE
      LOGICAL TOOBIG,ABORT
      REAL C(18)
      COMMON X(1000),LCTNLE,LCTNRE,CSTORE(18,60)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /COMP/ LCTNX(18)
C
      DO 10 OLDLE=1,NPLUS1
   10 CSTORE(OLDLE,NINT)=C(OLDLE)
      OLDLE=LCTNLE
      OLDRE=LCTNRE
      LCTNRE=MAXNUM
      LCTNLE=MIN0(MAXNUM-LNGTH+1,(MAXNUM-OLDLE+1)/2)-1
   20 LCTNLE=LCTNLE+1
      IF (MAXNUM-LCTNLE+1.LT.LNGTH) GO TO 40
      CALL SETP (CSTORE(1,NINT),X(LCTNLE),NLSMTH)
      NUMPTS=LCTNRE-LCTNLE+1
      CALL REMES (C,LCTNX,TOOBIG,ABORT)
      IF (ABORT) RETURN
      IF (TOOBIG) GO TO 20
      CALL STORE (CSTORE(1,NINT),OLDLE,LCTNLE)
      NINT=NINT+1
      DO 30 OLDLE=1,NPLUS1
   30 CSTORE(OLDLE,NINT)=C(OLDLE)
      RETURN
   40 ABORT=.TRUE.
      RETURN
C
      END


      SUBROUTINE STORE (C,LCTNLE,LCTNRE)
C
C  THIS SUBROUTINE OUTPUTS THE COEFFICIENTS AND ENDPOINTS OF THE
C  CURRENT APPROXIMATION AND SUBINTERVAL.  APPROPRIATE INFORMATION
C  IS STORED IN THE ARRAY CSTORE TO ALLOW THE ENTIRE PIECEWISE POLY-
C  NOMIAL APPROXIMATION TO BE EASILY EVALUATED AT ANY POINT BY THE
C  FUNCTION EVAL.
C
      DIMENSION C(18)
      COMMON XTABLE(500),FTABLE(502),CSTORE(18,60),DUM1(500)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
C
      NUMPTS=LCTNRE-LCTNLE+1
      WRITE (6,30) NINT,XTABLE(LCTNLE),XTABLE(LCTNRE),NUMPTS
      WRITE (6,40) (I,C(I),I=1,N)
      ERR=0.0
```

```
      DO 10 I=LCTNLE,LCTNRE
         TEMP=ABS(FTABLE(I)-HORNER(C,XTABLE(I),N))
         IF (TEMP.GT.ERR) ERR=TEMP
   10 CONTINUE
      WRITE (6,50) ERR
      DO 20 I=1,N
   20 CSTORE(I,NINT)=C(I)
      CSTORE(NPLUS1,NINT)=XTABLE(LCTNLE)
      RETURN
C
   30 FORMAT (//,5X, 15HINTERVAL NUMBER,I4, 16H WHICH BEGINS AT,E23.16,/
     1, 12H AND ENDS AT,E23.16,2X,  8HCONTAINS,I4,  8H POINTS.,/,  9H TH
     2E COEF, 51HFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE,/)
   40 FORMAT ((10X,  2HC(,I2,  3H) =,E24.16))
   50 FORMAT (/, 47H THE ERROR OF APPROXIMATION IN THIS INTERVAL IS,E24.
     116,  1H.)
C
      END


      SUBROUTINE DERIV (C,N)
C
C  THIS SUBROUTINE REPLACES THE COEFFICIENTS OF A POLYNOMIAL IN STAND-
C  ARD FORM WITH THE COEFFICIENTS OF THIS POLYNOMIAL-S DERIVATIVE.
C  THE NUMBER OF COEFFICIENTS, N, IS DECREMENTED.
C
      DIMENSION C(N)
C
      N=N-1
      DO 10 I=1,N
   10 C(I)=FLOAT(I)*C(I+1)
      RETURN
C
      END


      SUBROUTINE SETP (C,X,SMTH)
C
C  THIS SUBROUTINE APPROPRIATELY STORES IN THE ARRAY PPRIME THE VAL-
C  UES WHICH MUST BE INTERPOLATED TO GIVE THE PIECEWISE POLYNOMIAL THE
C  DESIRED SMOOTHNESS.
C
      DIMENSION C(18)
      COMMON /COMP/ CSTORE(18)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /DDIF/ PPRIME(5),DUM1(36)
      INTEGER SMTH
      DO 10 I=1,N
   10 CSTORE(I)=C(I)
      NDUMMY=N
      I=0
   20 IF (I.GT.SMTH) RETURN
      IF (I.EQ.0) GO TO 30
      CALL DERIV (CSTORE,NDUMMY)
   30 PPRIME(I+1)=HORNER(CSTORE,X,NDUMMY)
      I=I+1
      GO TO 20
C
      END


      FUNCTION CMPR(C,N,NEWTRY,OK)
```

```
C
C  THIS SUBROUTINE COMPARES THE FIRST DERIVATIVE OF THE CURRENT PIECE OF
C  THE PIECEWISE POLYNOMIAL APPROXIMATION EVALUATED AT XTABLE(NEWTRY)
C  WITH THE FIRST DERIVATIVE OF THE QUADRATIC INTERPOLATION OF F, CEN-
C  TERED AROUND XTABLE(NEWTRY), EVALUATED AT XTABLE(NEWTRY).  IF THESE
C  TWO DIFFER IN ABSOLUTE VALUE BY LESS THAN TOLER (EITHER ABSOLUTELY OR
C  RELATIVELY), WE SET OK TO .TRUE. AND WE ACCEPT XTABLE(NEWTRY) AS A
C  REASONABLE SUBINTERVAL RIGHT ENDPOINT.  NOTE THAT THE USER MAY EASILY
C  CHANGE TOLER BY MEANS OF THE FOLLOWING DATA STATEMENT.
C
      LOGICAL OK
      COMMON XTABLE(500),FTABLE(500),DUM1(1582)
      DIMENSION C(18)
      DATA TOLER/.01/
C
      OK=.FALSE.
      A=(FTABLE(NEWTRY)-FTABLE(NEWTRY-1))/(XTABLE(NEWTRY)-XTABLE(NEWTRY-
     11))
      B=(FTABLE(NEWTRY+1)-FTABLE(NEWTRY))/(XTABLE(NEWTRY+1)-XTABLE(NEWTR
     1Y))
      D=(B-A)/(XTABLE(NEWTRY+1)-XTABLE(NEWTRY-1))
      A=A+D*(XTABLE(NEWTRY)-XTABLE(NEWTRY-1))
      B=HORNER(C,XTABLE(NEWTRY),N)
      CMPR=ABS(A-B)
      IF (CMPR.LE.TOLER) OK=.TRUE.
      A=ABS(A)
      IF (A.LT..01) RETURN
      IF (CMPR/A.LT.TOLER) OK=.TRUE.
      RETURN
C
      END


      SUBROUTINE REMES (C,LCTNX,TOOBIG,ABORT)
C
C  THIS IS THE DRIVING PROGRAM FOR THE COMPUTATION OF THE BEST
C  RESTRICTED RANGE UNIFORM POLYNOMIAL APPROXIMATION TO F(X) (VALUES
C  ARE STORED IN XTABLE AND FTABLE) OF DEGREE LESS THAN OR EQUAL TO N-1
C  ON THE SUBINTERVAL [XTABLE(LCTNLE),XTABLE(LCTNRE)].  SEE THE PAPER BY
C  B. CHALMERS FOR ADDITIONAL INFORMATION ON THIS ALGORITHM.
C
      DIMENSION XTPTS(18), C(18), SGNRXI(18), LCTNX(18), LCTNZ(18)
      COMMON XTABLE(500),FTABLE(500),LCTNLE,DUMMY(1081),ERROR(500),FU(50
     10),FL(500)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT
      COMMON /DDIF/ DUM1(23),D(18)
      LOGICAL STOP,TOOBIG,ABORT
      INTEGER FRSTM1,START
C
C  EPS IS A MACHINE CONSTANT--SET EPS TO (APPROXIMATELY) THE SMALLEST
C  VALUE SUCH THAT EPS + 1.0 IS GREATER THAN 1.0.
C
      DATA ITERMX,EPS/30,1.0E-10/
C
C  FIRST WE INITIALIZE VARIOUS ARRAYS AND VALUES.
C
      TOOBIG=.FALSE.
      FRSTM1=LCTNLE-1
      SGNRXI(1)=1.0
      DO 10 I=2,NX
   10 SGNRXI(I)=-SGNRXI(I-1)
      NFPTS=NUMGRD-2
```

```
      START=2
      ERROR(1)=0.0
      IF (NLSMTH.GE.0) GO TO 20
      NFPTS=NFPTS+1
      START=1
   20 IF (NRSMTH.LT.0) NFPTS=NFPTS+1
      DELTA=FLOAT(NFPTS-1)/FLOAT(NX-1)
      DO 30 I=1,NX
         LCTNX(I)=START+IFIX(FLOAT(I-1)*DELTA+.5)
         J=FRSTM1+LCTNX(I)
         XTPTS(I)=XTABLE(J)
         D(I)=FTABLE(J)
   30 CONTINUE
C
C     NOW WE BEGIN ITERATING.  CONVERGENCE OCCURS WHEN TWO CONSECUTIVE
C     REFERENCE SETS (DETERMINED BY SOLVE) ARE THE SAME.
C
      DO 70 ITER=1,ITERMX
         CALL DIVDIF (C,LCTNX,SGNRXI,ABORT)
         IF (ABORT) RETURN
         DO 40 I=START,NUMGRD
   40    ERROR(I)=FTABLE(I+FRSTM1)-BPOLY(XTABLE(I+FRSTM1),C,N)
         IF (ABS(C(NPLUS1)).LE.EPS) GO TO 60
         DO 50 I=1,NX
            IF (SGNRXI(I).NE.0.0) SGNRXI(I)=SIGN(1.0,ERROR(LCTNX(I)))
            IF (SGNRXI(I).EQ.0.0) SGNRXI(I)=FLOAT(LCTNZ(I))
            IF (I.EQ.1) GO TO 50
            IF (SGNRXI(I)*SGNRXI(I-1).GE.0.0) GO TO 80
   50    CONTINUE
   60    CALL ZEROFD (LCTNX,LCTNZ,SGNRXI,ERROR)
         CALL SOLVE (XTPTS,LCTNX,LCTNZ,SGNRXI,ABS(C(NPLUS1)),STOP,TOOBIG
     1   )
         IF (.NOT.STOP) GO TO 70
         IF (.NOT.TOOBIG) CALL TRANS (C,N)
         RETURN
   70 CONTINUE
C
C     WE PRINT OUT ERROR MESSAGES IF SOMETHING GOES WRONG.
C
      WRITE (6,100) ITERMX
      GO TO 90
   80 WRITE (6,110) ITER
   90 ABORT=.TRUE.
      RETURN
C
  100 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X,40HTHE REMES ALGORITHM HAS NOT
     1 CONVERGED IN,I3,12H ITERATIONS.,11X,1H*,/,2H0*,11X,36HPROGRAM ABO
     2RTED IN SUBROUTINE REMES.,30X,1H*,/,1H0,39(2H* ),1H*)
  110 FORMAT (1H0,39(2H* ),1H*,/,2H0*,8X,12HIN ITERATION,I3,47H OF REMES
     1. NO ALTERNATION OF SIGN OCCURS AT THE,7X,1H*,/,2H0*,8X,57HEXTREME
     2 POINTS.  THE PROGRAM ABORTED IN SUBROUTINE REMES.,12X,1H*,/,1H0,3
     39(2H* ),1H*)
C
      END


      SUBROUTINE DIVDIF (C,LCTNX,SGNRXI,ABORT)
C
C     THIS SUBROUTINE MAKES USE OF A DIVIDED DIFFERENCE SCHEME FOR SOLVING
C     THE VANDERMONDE-LIKE LINEAR SYSTEM INHERENT IN THE REMES ALGORITHM.
C     THE ADVANTAGES OF USING THIS SPECIAL PURPOSE LINEAR SYSTEM SOLVER
C     ARE--
```

BEST AVAILABLE COPY

```
C
C     THIS ROUTINE REQUIRES ON THE ORDER OF N**2 OPERATIONS AS COMPARED
C     TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**3 OPER-
C     ATIONS.
C
C     THIS ROUTINE REQUIRES ON THE ORDER OF N STORAGE LOCATIONS AS COM-
C     PARED TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER OF N**2.
C
C  SEE THE FORTHCOMING PAPER BY J. A. HULL, S. F. MCCORMICK, AND G. D.
C  TAYLOR FOR A COMPLETE DESCRIPTION OF THIS ALGORITHM.
C
      COMMON XTABLE(500),FTABLE(500),LCTNLE,LCTNRE,CSTORE(18,60),ERROR(5
     100)
      COMMON /DDIF/ PPRIME(5),X(18),D(18)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      DIMENSION LCTNX(18), C(18), SGNRXI(18)
      INTEGER FRSTM1
      LOGICAL ABORT
C
C  FIRST WE INITIALIZE SEVERAL VARIABLES.
C
      FRSTM1=LCTNLE-1
      IF (NRSMTH.GE.0) SGNRXI(NPLUS1)=0.0
      IF (NRSMTH.LT.0) SGNRXI(NPLUS1)=SGNRXI(NX)
C
C  SET UP THE VECTOR X AND THE FIRST ROW OF THE DIVIDED DIFFERENCE TAB-
C  LE, USING THE COEFFICIENT VECTOR C FOR TEMPORARY STORAGE.
C
      I=0
   10 IF (I.GT.NLSMTH) GO TO 20
      I=I+1
      X(I)=XTABLE(LCTNLE)
      C(I)=PPRIME(1)
      GO TO 10
   20 J=0
   30 IF (J.GE.NX) GO TO 40
      I=I+1
      J=J+1
      X(I)=XTABLE(FRSTM1+LCTNX(J))
      C(I)=D(J)
      GO TO 30
   40 IF (I.GE.NPLUS1) GO TO 50
      I=I+1
      X(I)=XTABLE(LCTNRE)
      C(I)=PPRIME(NLSMTH+2)
      GO TO 40
   50 CONTINUE
C
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.
C
      FAC=1.0
      DO 100 J=2,N
         JP1=J+1
         JM1=J-1
         FAC=FAC*FLOAT(JM1)
         TEMP2=C(JM1)
         DO 90 I=J,N
            IF (X(I).NE.X(I-JM1)) GO TO 70
            IF (I.GT.NLSMTH+1) GO TO 60
            IF (J.GT.NPLUS1-NX) GO TO 220
            TEMP1=PPRIME(J)/FAC
            GO TO 80
```

```
   60       IF (NLSMTH+JP1.GT.NPLUS1-NX) GO TO 220
            TEMP1=PPRIME(NLSMTH+JP1)/FAC
   70       TEMP1=(C(I)-C(I-1))/(X(I)-X(I-JM1))
   80       C(I-1)=TEMP2
            TEMP2=TEMP1
   90    CONTINUE
         C(N)=TEMP2
  100 CONTINUE
C
C  L**(-1)F HAS NOW BEEN TEMPORARILY STORED IN THE COEFFICIENT ARRAY C.
C  WE NOW COMPUTE L**(-1)C.  WE WILL COMPUTE THE DIVIDED DIFFERENCES
C  IN THE TEMPORARY STORAGE ARRAY D.  FIRST WE SET UP THE FIRST COLUMN
C  OF THE DIVIDED DIFFERENCE TABLE.
C
      I=0
  110 IF (I.GT.NLSMTH) GO TO 120
      I=I+1
      D(I)=0.0
      GO TO 110
  120 J=0
  130 IF (J.GE.NX) GO TO 140
      I=I+1
      J=J+1
      D(I)=SGNRXI(J)
      GO TO 130
  140 IF (I.GE.N) GO TO 150
      I=I+1
      D(I)=0.0
      GO TO 140
  150 CONTINUE
C
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFEPENCES.
C
      DO 190 J=2,N
         TEMP2=D(J-1)
         DO 180 I=J,N
            IF (X(I).NE.X(I-J+1)) GO TO 160
            TEMP1=0.0
            GO TO 170
  160       TEMP1=(D(I)-D(I-1))/(X(I)-X(I-J+1))
  170       D(I-1)=TEMP2
            TEMP2=TEMP1
  180    CONTINUE
         D(N)=TEMP2
  190 CONTINUE
C
C  WE NOW COMPUTE M=(F(N+1)-W[TRANSPOSE]*B1))/(0.0-W[TRANSPOSE]*B2)
C  =C(N+1)=(UNIFORM ERROR)
C  FIRST WE COMPUTE THE TWO DOT PRODUCTS SIMULTANEOUSLY.
C
      W=1.0
      B1=0.0
      B2=0.0
      DO 200 I=1,N
         B1=B1+C(I)*W
         B2=B2+D(I)*W
         W=W*(X(NPLUS1)-X(I))
  200 CONTINUE
C
C  NOW WE COMPUTE M
C
      C(NPLUS1)=(C(NPLUS1)-B1)/(SGNRXI(NPLUS1)-B2)
```

```
C
C  FINALLY, WE COMPUTE THE COEFFICIENTS C(I).
C
      DO 210 I=1,N
         C(I)=C(I)-C(NPLUS1)*D(I)
  210 CONTINUE
      RETURN
  220 ABORT=.TRUE.
      WRITE (6,230)
      RETURN
C
  230 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X,54HSUBROUTINE DIVDIF HAS FAILE
     1D--PROBABLY DUE TO AN INPUT,12X,1H*,/,2H0*,11X,58HERROR (TO DIVDIF
     2)--NOT ENOUGH ELEMENTS IN THE ARRAY APRIME,8X,1H*,/,2H0*,11X,56HOR
     3 TWO IDENTICAL POINTS IN THE ARRAY X.  PROGRAM ABORTED,10X,1H*,/,2
     4H0*,11X,21HIN SUBROUTINE DIVDIF.,45X,1H*,/,1H0,39(2H* ),1H*)
C
      END


      SUBROUTINE ZEROFD (LCTNX,LCTNZ,SGNRXI,ERROR)
C
C  THIS SUBROUTINE LOCATES THE (APPROXIMATE) ZEROS BETWEEN CONSECUTIVE
C  POINTS OF THE CURRENT REFERENCE SET.  THE LOCATIONS OF THESE POINTS
C  IN THE ARRAY XTABLE RELATIVE TO THE BEGINNING OF THE CURRENT SUB-
C  INTERVAL ARE STORED IN LCTNZ.
C
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT
      DIMENSION LCTNX(18), LCTNZ(18), SGNRXI(18), ERROR(300)
C
      LCTNZ(1)=MIN0(2,2+NLSMTH)
      LCTNZ(NX+1)=MAX0(NUMGRD-1,NUMGRD-1-NRSMTH)
      DO 30 I=2,NX
         LTNIM1=LCTNX(I-1)
         NUMPTS=LCTNX(I)-LTNIM1
         DO 10 J=1,NUMPTS
            NEWTRY=LTNIM1+J
            IF (ERROR(LTNIM1)*ERROR(NEWTRY).LE.0.0) GO TO 20
   10    CONTINUE
   20    LCTNZ(I)=NEWTRY
   30 CONTINUE
      RETURN
C
      END


      SUBROUTINE SOLVE (X,LCTNX,LCTNZ,SGNRXI,E,STOP,TOOBIG)
C
C  THIS SUBROUTINE PERFORMS THE MULTIPLE EXCHANGE OF THE REFERENCE
C  SET REPUIRED AT EACH ITERATION OF THE REMES ALGORITHM.
C
      COMMON XTABLE(500),FTABLE(500),LCTNLE,DUMMY(1081),ERROR(500),FU(50
     10),FL(500)
      COMMON /DDIF/ DUM1(23),D(18)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,DUM(4)
      DIMENSION X(18), LCTNX(18), LCTNZ(18), SGNRXI(18)
      LOGICAL STOP,TOOBIG
      INTEGER FRSTM1,RTEND
C
C  EPS IS A MACHINE CONSTANT--SET EPS TO ROUGHLY THE SMALLEST NUMBER
C  SUCH THAT 1.0 + EPS .GT. 1.0.
C
```

```
      DATA EPS/1.0E-9/
C
      STOP=.TRUE.
      FRSTM1=LCTNLE-1
C
C  WE FIRST COMPUTE THE LOCATIONS OF THE NEW SET OF EXTREME POINTS,
C  STORING THEM IN THE VECTOR LCTNX.  WE BEGIN BY CHOOSING AS THE ITH
C  ELEMENT OF LCTNX THE LOCATION OF THE GRIDPOINT IN THE SUBINTERVAL
C  BETWEEN THE ITH AND (I+1)ST ZERO WHICH RESULTS IN THE LARGEST ERROR
C  OF THE SAME SIGN AS THE PREVIOUS ITH EXTREME POINT (THEREBY GUARANT-
C  EEING ALTERNATION).  AT THE SAME TIME WE SEARCH FOR THE GRIDPOINT
C  WHICH RESULTS IN THE LARGEST (ABSOLUTE) ERROR, STORING ITS LOCATION
C  (IN LNBGST) AND THE NUMBER OF THE SUBINTERVAL IN WHICH IT OCCURS (IN
C  INBGST).
C
      BIGER=-1.0E30
      BIGEST=-1.0E30
      DO 70 INTNUM=1,NX
         BIG=-1.0E30
         LFTEND=LCTNZ(INTNUM)
         RTEND=LCTNZ(INTNUM+1)
         SGN=SGNRXI(INTNUM)
         DO 60 NEWLOC=LFTEND,RTEND
            SAME1=SGN*ERROR(NEWLOC)-E
            OPP1=-SGN*ERROR(NEWLOC)-E
            IF (SGN.LT.0.0) GO TO 10
            SAME2=ERROR(NEWLOC)-FL(FRSTM1+NEWLOC)
            OPP2=-ERROR(NEWLOC)-FU(FRSTM1+NEWLOC)
            GO TO 20
 10         SAME2=-ERROR(NEWLOC)-FU(FRSTM1+NEWLOC)
            OPP2=ERROR(NEWLOC)-FL(FRSTM1+NEWLOC)
 20         IF (SAME1.LE.BIG) GO TO 30
            BIG=SAME1
            LCTNX(INTNUM)=NEWLOC
            LCTNZ(INTNUM)=0
 30         IF (SAME2.LE.BIG) GO TO 40
            BIG=SAME2
            LCTNX(INTNUM)=NEWLOC
            LCTNZ(INTNUM)=IFIX(SGN)
 40         IF (BIGER.LT.BIG) BIGER=BIG
            IF (BIGEST.LT.BIG) BIGEST=BIG
            IF (OPP1.LE.BIGEST) GO TO 50
            BIGEST=OPP1
            INBGST=INTNUM
            LNBGST=NEWLOC
            KIND=0
 50         IF (OPP2.LE.BIGEST) GO TO 60
            BIGEST=OPP2
            INBGST=INTNUM
            LNBGST=NEWLOC
            KIND=-IFIX(SGN)
 60      CONTINUE
 70   CONTINUE
      IF (BIGEST.LT.EPS) RETURN
      IF (ABS(BIGER-BIGEST).LT.EPS) GO TO 120
C
C  AT THIS POINT IT IS STILL NECESSARY TO INSERT THE LOCATION OF THE
C  GRIDPOINT RESULTING IN THE LARGEST ERROR INTO LCTNX.  THERE ARE THREE
C  CASES, EACH OF WHICH IS HANDLED SEPARATELY--ALTERNATION IS PRESERVED
C  AT THE GRIDPOINTS.
C
      YBIGST=XTABLE(FRSTM1+LNBGST)
```

```
      YIBIG=XTABLE(LCTNX(INBGST)+FRSTM1)
      IF ((INBGST.EQ.1).AND.(YBIGST.LT.YIBIG)) GO TO 80
      IF ((INBGST.EQ.NX).AND.(YBIGST.GT.YIBIG)) GO TO 100
      IF (YBIGST.LT.YIBIG) NEWINT=INBGST-1
      IF (YBIGST.GT.YIBIG) NEWINT=INBGST+1
      LCTNX(NEWINT)=LNBGST
      LCTNZ(NEWINT)=KIND
      GO TO 120
   80 DO 90 I=2,NX
         J=NX-I+2
         LCTNX(J)=LCTNX(J-1)
         LCTNZ(J)=LCTNZ(J-1)
         SGNRXI(J)=SGNRXI(J-1)
   90 CONTINUE
      LCTNX(1)=LNBGST
      LCTNZ(1)=KIND
      IF (KIND.EQ.0) SGNRXI(1)=-SGNRXI(1)
      GO TO 120
  100 DO 110 J=1,NXM1
         LCTNX(J)=LCTNX(J+1)
         LCTNZ(J)=LCTNZ(J+1)
         SGNRXI(J)=SGNRXI(J+1)
  110 CONTINUE
      LCTNX(NX)=LNBGST
      LCTNZ(NX)=KIND
      IF (KIND.EQ.0) SGNRXI(NX)=-SGNRXI(NX)
C
C  NOW THAT LCTNX IS ACCEPTABLE, WE CHECK FOR CONVERGENCE OF THE ALGOR-
C  ITHM. STORING THE EXTREME POINTS IF THE CONVERGENCE CRITERION IS NOT
C  MET.
C
  120 DO 130 I=1,NX
         IF (LCTNZ(I).EQ.0) GO TO 140
  130 CONTINUE
      TOOBIG=.TRUE.
      RETURN
  140 DO 170 I=1,NX
         NEWLOC=FRSTM1+LCTNX(I)
         IF (LCTNZ(I).NE.0) GO TO 150
         D(I)=FTABLE(NEWLOC)
         GO TO 160
  150    IF (LCTNZ(I).EQ.1) D(I)=FTABLE(NEWLOC)-FL(NEWLOC)
         IF (LCTNZ(I).EQ.-1) D(I)=FTABLE(NEWLOC)+FU(NEWLOC)
         SGNRXI(I)=0.0
  160    TEMP=XTABLE(NEWLOC)
         IF (ABS(TEMP-X(I)).LE.EPS) GO TO 170
         X(I)=TEMP
         STOP=.FALSE.
  170 CONTINUE
      RETURN
C
      END


      FUNCTION BPOLY(XX,C,N)
C
C  THIS FUNCTION IS USED TO EVALUATE (BY THE APPROPRIATE ADAPTATION OF
C  HORNERS METHOD) THE POLYNOMIAL
C
C     C(1) + C(2)*(XX-X(1)) + C(3)*(XX-X(1))*(XX-X(2)) + . . .
C        + C(N)*(XX-X(1))*(XX-X(2))* . . . *(XX-X(N-1))
C
```

```
      DIMENSION C(18)
      COMMON /DDIF/ DUMMY(5),X(18),DUM(18)
C
      NM1=N-1
      BPOLY=C(N)
      DO 10 I=1,NM1
         J=N-I
         BPOLY=C(J)+(XX-X(J))*BPOLY
   10 CONTINUE
      RETURN
C
      END


      SUBROUTINE TRANS (C,N)
C
C  THIS SUBROUTINE TRANSFORMS A POLYNOMIAL WRITTEN IN THE FORM
C
C      C(1) + C(2)*(X-X(1)) + C(3)*(X-X(1))*(X-X(2)) + . . .
C        + C(N)*(X-X(1))*(X-X(2))* . . . *(X-X(N-1))
C
C  TO A POLYNOMIAL WRITTEN IN TERMS OF POWERS OF X.  THE X(I)-S ARE
C  SUPPLIED BY SUBROUTINE DIVDIF.
C
      DIMENSION C(18)
      COMMON /DDIF/ DUMMY(5),X(18),DUM(18)
C
      NM1=N-1
      DO 20 J=1,NM1
         K=N-J
         DO 10 I=K,NM1
            C(I)=C(I)-X(K)*C(I+1)
   10    CONTINUE
   20 CONTINUE
      RETURN
C
      END


      FUNCTION EVAL(X)
C
C  THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION AT
C  ANY POINT IN THE ENTIRE INTERVAL.
C
      COMMON DUMMY(1002),CSTORE(18,60),DUM(500)
      COMMON /SCALAR/ N,NPLUS1,DUM2(5),NINT
      IF (NINT.LT.2) GO TO 20
      DO 10 I=2,NINT
         ISTORE=I-1
         IF (X.LE.CSTORE(NPLUS1,I)) GO TO 30
   10 CONTINUE
      ISTORE=NINT
      GO TO 30
   20 ISTORE=1
   30 EVAL=HORNER(CSTORE(1,ISTORE),X,N)
      RETURN
C
      END


      FUNCTION HORNER(C,X,N)
C
```

```
C  THIS FUNCTION EVALUATES A POLYNOMIAL IN STANDARD FORM BY HORNERS      HORNER30
C  METHOD.                                                               HORNER40
C                                                                        HORNER50
      DIMENSION C(N)                                                     HORNER60
C                                                                        HORNER70
      HORNER=C(N)                                                        HORNER80
      I=N                                                                HORNER90
   10 IF (I.LT.2) RETURN                                                 HORNE100
      HORNER=HORNER*X+C(I-1)                                             HORNE110
      I=I-1                                                              HORNE120
      GO TO 10                                                           HORNE130
C                                                                        HORNE140
      END                                                                HORNE150
```

RESTRICTED RANGE ADAPTIVE CURVE FITTING PROGRAM : SAMPLE RUN
(ALGORITHMICALLY DEFINED RESTRAINING CURVES)

INPUT :

```
1                              ( THIS DENOTES RESTRAINTS OPTION )
6     2     4.00     .175      ( N, SMTH, MAXTOL, MINTOL )
                               ( XTABLE, FTABLE)
3.0        0.0
5.0        1.3
7.0        3.4
11.0       5.2
13.0       6.0
15.0      14.4
17.5      21.4
20.0      27.4
22.5      50.9
25.0      49.3
27.5      47.5
30.0      51.5
35.0      36.5
40.0      27.9
50.0       9.4
60.0       4.2
```

TO EMPLOY THE USER DEFINED RESTRAINING CURVES OPTION, THE FIRST DATA CARD SHOULD BE 0, AND THE UPPER AND LOWER TOLERANCES TO BE ALLOWED AT EACH DATA POINT SHOULD BE ON THE RESPECTIVE DATA CARDS.
EX:

```
27.5     47.5     1.0     4.0
                  (U)     (L)
```

OUTPUT :

INTERVAL NUMBER 1 WHICH BEGINS AT .3000000000000000E+01 AND ENDS AT .1100000000000000E+02 CONTAINS 41 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .1497565266967138E+02
C( 2) = -.1144565861352237E+02
C( 3) =  .3049103763268548E+01
C( 4) = -.3346122135351450E+00
C( 5) =  .1592800176610654E-01
C( 6) = -.2534785932326346E-03
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .3251703402472117E+00.

INTERVAL NUMBER 2 WHICH BEGINS AT .1100000000000000E+02 AND ENDS AT .1500000000000000E+02 CONTAINS 24 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .8953993229386797E+03
C( 2) = -.3853784676856376E+03
C( 3) =  .6639335801960442E+02
C( 4) = -.5682948501274041E+01
C( 5) =  .2409284901711972E+00
C( 6) = -.4025042180238581E-02
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1207492591612606E+01.

INTERVAL NUMBER 3 WHICH BEGINS AT .1500000000000000E+02
AND ENDS AT .2000000000000000E+02 CONTAINS 29 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .2080819121698872E+05
C( 2) = -.6119991900236899E+04
C( 3) =  .7147141380302528E+03
C( 4) = -.4142494450662980E+02
C( 5) =  .1192134278058383E+01
C( 6) = -.1362678361482650E-01
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2829293208918557E+01.

INTERVAL NUMBER 4 WHICH BEGINS AT .2000000000000000E+02
AND ENDS AT .2359270706355915E+02 CONTAINS 28 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.2671258659287486E+06
C( 2) =  .6180344279850274E+05
C( 3) = -.5704535104277922E+04
C( 4) =  .2625284927906778E+03
C( 5) = -.6022618197926448E+01
C( 6) =  .5509132822059026E-01
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .3398666798696013E+01.

INTERVAL NUMBER 5 WHICH BEGINS AT .2359270706355915E+02
AND ENDS AT .2738518411304983E+02 CONTAINS 21 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.1013104278693695E+07
C( 2) =  .2002989828887461E+06
C( 3) = -.1582334382597386E+05
C( 4) =  .6243710376956733E+03
C( 5) = -.1230580964377970E+02
C( 6) =  .9691424185808195E-01
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .7623136251345386E+00.

INTERVAL NUMBER 6 WHICH BEGINS AT .2738518411304983E+02
AND ENDS AT .3343698511741786E+02 CONTAINS 37 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .2974023810520601E+06
C( 2) = -.4737982327455143E+05
C( 3) =  .3008544262538955E+04
C( 4) = -.9516814694657342E+02
C( 5) =  .1499809050599829E+01
C( 6) = -.9421877586332617E-02
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1270142945357293E+01.

INTERVAL NUMBER 7 WHICH BEGINS AT .3343698511741786E+02
AND ENDS AT .4085087038553866E+02 CONTAINS 39 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .6567620303936116E+05
C( 2) = -.8417321059978451E+04
C( 3) =  .4305390572534816E+03
C( 4) = -.1097600576167463E+02
C( 5) =  .1394356985419112E+00
C( 6) = -.7061442077258562E-03
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1270142944891631E+01.

INTERVAL NUMBER 8 WHICH BEGINS AT .4085087038553866E+02
AND ENDS AT .6000000000000000E+02 CONTAINS 95 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .1984768817575497E+05
C( 2) = -.2082281676169310E+04
C( 3) =  .8723916835701220E+02
C( 4) = -.1819359280069307E+01
C( 5) =  .1886185858999134E-01
C( 6) = -.7772177593868418E-04
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .7764477640884024E+00.